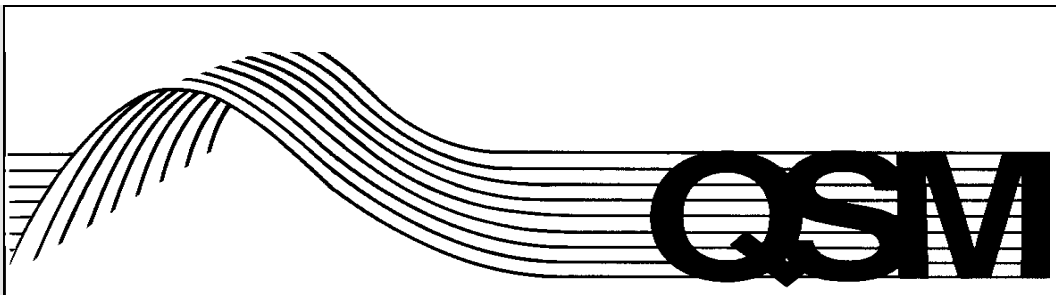

MEASURES FOR EXCELLENCE

AVOIDING

THE PREMATURE

DELIVERY

OF SOFTWARE

J.W.E Greene
QUANTITATIVE SOFTWARE MANAGEMENT LTD

7 Rue Fenoux
75015 Paris
Tel : 33-140-431210
Fax : 33-148-286249

Internet : qsm.europe@pobox.com
Compuserve : 100113,3364
www.qsm.com

93 Blythe Road
London W14 0HP
Tel : 44-171-603-9009
Fax : 44-171-602-6008

1. Introduction

Premature delivery of software is characterised by high levels of outstanding defects that become apparent when the software is used. Commercial pressure increasingly leads to attempts to deliver software before defects are found and fixed.

The commercial pressure arises from :

- fixed price software development contracts that exceed the agreed price
- management decisions to release software regardless of completing the outstanding test program
- fixed dates for product delivery to meet external deadlines such as homologation

Inevitably when such pressures lead to premature delivery the latent defects then emerge and the consequences become apparent. Frequently the effort to respond is highly disruptive and leads to more cost and lost time than following a sensible plan to achieve the required reliability. Figure 1 illustrates the concept. This is based on the first case study.

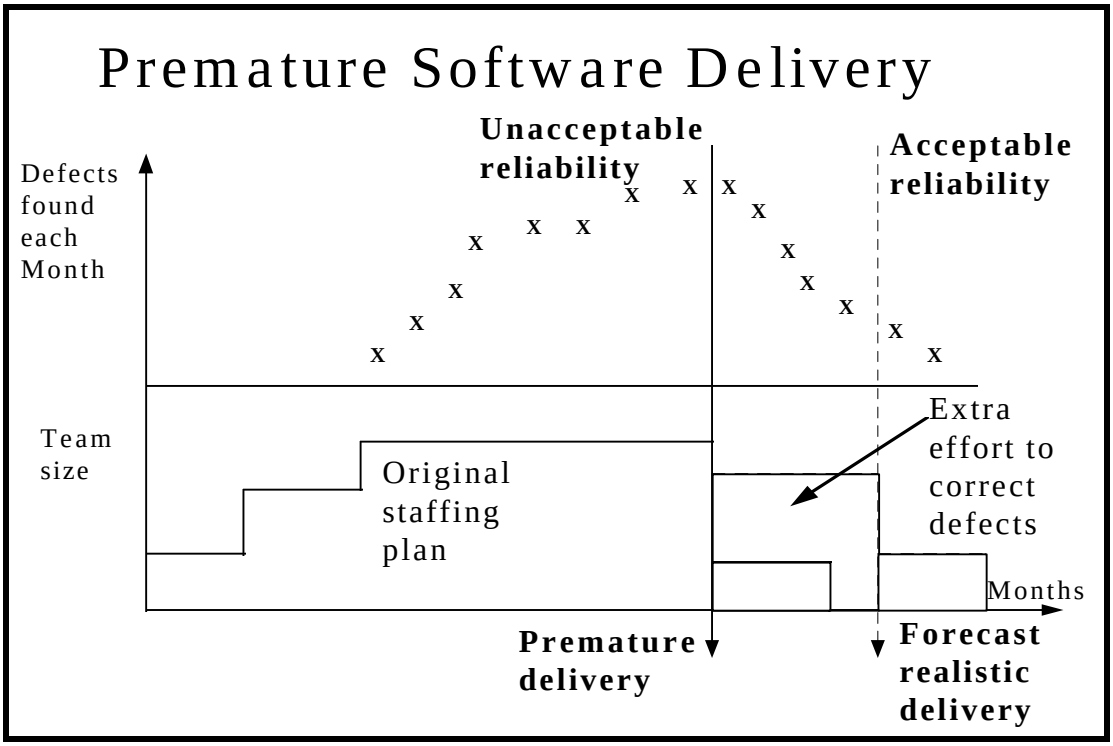

Figure 1 : Illustration of Premature Software Delivery.

Our work tracking and controlling software development shows how such situations are detected and prevented. Prevention depends on determining the number of outstanding defects and ensuring testing continues to meet

reliability goals. This may also mean evaluating alternative strategies to meet rigid deadlines.

The key to prevention is visibility throughout development. In particular the continuous visibility of the software defect levels and forecasts of latent errors.

Visibility is particularly important to purchasers of high technology software based systems. These systems include those particularly sensitive to failure such as real time control, avionics, telecommunications as well as many military developments. Here the purchasing organisation needs to be confident that delivery of the final software product is made at a time when there are few errors and a long time between software failures.

The concept of time between failures, technically termed “Mean Time To Defect MTTD”, is now a formal criteria imposed on software developers. For instance the Royal Netherlands PTT (Ref. 1) defines software acceptance milestones in terms of meeting explicit MTTD criteria.

2. Objectives

The objectives in this paper are to highlight the problems associated with premature software delivery and to show how these are detected and prevented. Case studies show how to gain the key objectives wherever there is a concern to develop and take delivery of highly reliable software.

These key objectives are to:

- make visible and quantify the reliability of software as it is developed
- use reported defects to tune an empirical reliability model
- forecast the outstanding defects and MTTD
- determine explicit MTTD goals for acceptance and operation
- link contract milestone payments with these goals
- minimise “maintenance” costs

In this paper we a) describe the QSM model of defect behaviour b) the basic data needed c) show the practical results from applying these techniques and how the objectives are achieved.

3. The QSM Defect Model

The analysis of actual defect data from a large number of software projects shows that the errors can be modelled using a form of Rayleigh model. (Ref.: 2) This represents well the iterative design processes in which significant feedback is inherently part of the solution process. Further, QSM research confirms that the Rayleigh reliability model closely approximates the actual empirical defect data collected over time from a large number of completed projects. Independent research from IBM confirms this behaviour (Ref. 7).

Analysis derived from many thousands of developments confirms the key drivers that determine software defects. The three fundamental drivers are :

1. the amount of software : as this increases the number of defects increase. The rate of defect increase is close to linear.

2. an independent measure of process productivity, the Productivity Index : PI. Here our empirical data shows that the number of defects decreases as the Productivity Index increases. The decrease is exponential.

3. the peak staffing : Simply put the more people who work on the development the more errors will be produced in the software product. Adding people increases the defect creation process at an accelerating rate.

In the QSM reliability modelling approach the Rayleigh equation is used to predict the number of defects discovered over time. A general (normalised) software error model is used that is tuned using the actual defects reported. In this way the model matches the defect discovery behaviour occurring in each development.

4. Data For Detecting and Preventing Premature Delivery

Our work with purchasing organisations involves tender evaluation of software bids that leads to the selection of a software development contractor. (Ref.: 1) The formal tender evaluation method we apply requires the contractor to set out their development plan, key milestones and the functionality and size of the proposed software. This provides a baseline against which development is tracked.

In the formal contract for the development the software developer is required to provide high level progress data typically every 2 or 4 weeks. The high level progress data used in the QSM model is fundamental to the overall project management of the development. As such it costs nothing to supply. (Ref. 4).

The regular reported progress data consists of :

- the number of staff on the project in each period
- key milestone dates such as the completion of coding and start of full integration
- the status of each sub-system (specification, design, coding, etc.)
- changes to the estimated size of each sub-system or the number of sub-systems
- the number of software defects found

Using this data it is practical to compare the progress to date against the baseline plan and detect significant variance. In particular the defects being found and reported are used to tune the error model to forecast the

outstanding software errors (Ref. 3). Normally this includes categorising errors based on their severity.

5. Case Study 1 : Fixed Delivery Dates

Here the software is developed as part of equipment that has to meet a strict deadline for homologation. In this case study, the software development gave cause for concern since high numbers of errors were reported after "delivery" in January 1995 to the hardware developers.

In January the software staff levels were reduced and people switched to new projects. This was reversed as the situation became clear with respect to the remaining errors. The resulting management confusion cost a great deal of time, money and energy.

Fortunately the software development organisation keep very good and detailed records of the errors found every two weeks. This allows the situation up to the date of premature delivery to be assessed using the error model. This "playback" of the position found in January then highlights the latent errors that remain.

To do this at the date of "January delivery" a forecast is made to estimate the number of outstanding errors. Continuous recording of errors after "delivery" allows the comparison between the forecast errors and actuals. This in turn provides an objective assessment of how well the model estimates the outstanding errors based on the data available in January.

5.1 Case Study 1 Results

In Figure 5 we show the forecast made using the error data recorded up to January 1995. At this point the forecast indicates that only approximately half the errors have been found against a forecast total of 680 and that a realistic delivery date is expected in April 1995.

In the event the actual cumulative defects reported up to the end of April 1995 amount to 720 defects.

Clearly the delivery of the software in January was premature given that only 50% of the total errors had been found and the rate of detection was still high. The reality of the situation rapidly became clear as the remaining errors manifested themselves.

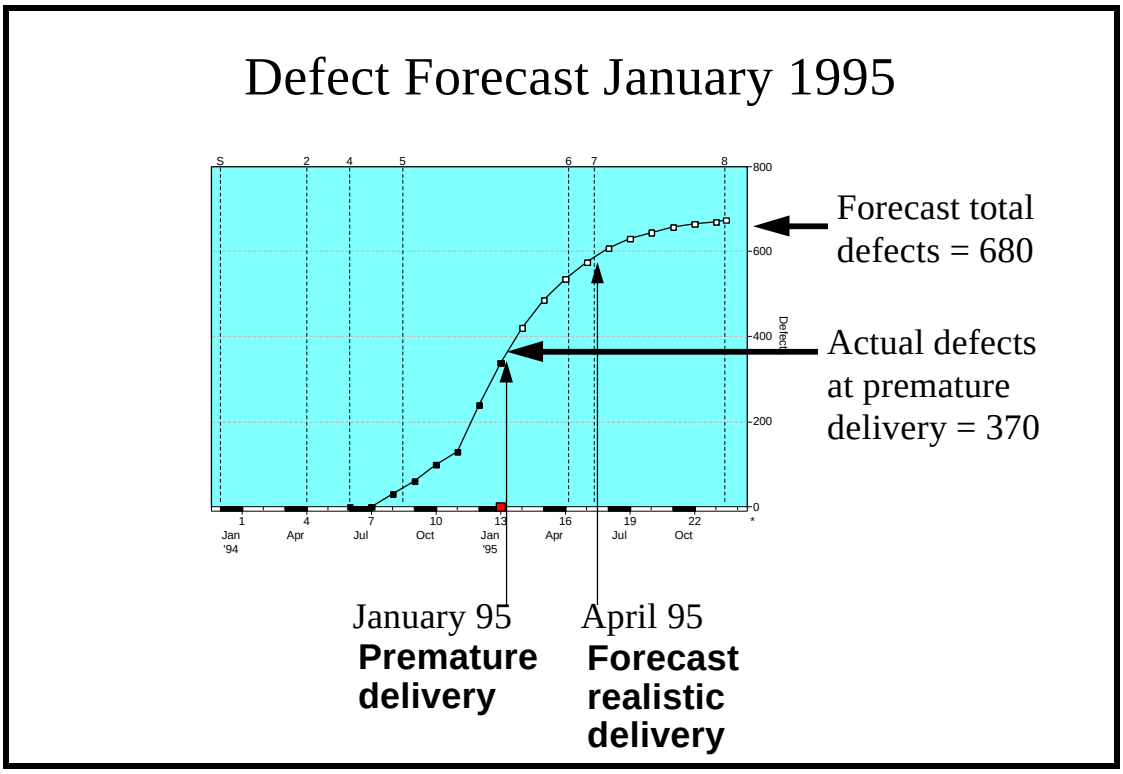

Figure 5 : Actual and Forecast Defects

Management would have been aware of the dangerous situation in allowing the premature delivery if the forecast been made in January using the available data. The realistic delivery date of April 1995 is forecast at which time the MTTD of the software is sufficiently good to permit integration with the hardware.

6. Case Study 2 : Fixed Price Software

Here the development is fixed price for a telecommunications system. The requirement is for high reliability with a delivery of software that achieves a high Mean Time To Defect.

The project in question was first analysed in January 1990 when the original planned completion data of November 1989 had already been missed. The situation at that time is described in detail by the author in an earlier publication showing how to get control of a run away project. (Ref. 5).

Subsequently QSM was requested to track progress in the development. The initial fixed price for this software was £400,000 with delivery promised by the contractor in November 1989. Our analysis in January 1990 forecast development costs of at least £2,000,000 with completion slipping by an additional 9 months to November 1990. Subsequently the progress data reported each month revealed the situation was even more serious in terms of cost overrun and slippage.

In such a situation the commercial pressure on a contractor to cut losses becomes enormous. This is attempted in three possible ways :

- cut staff and re-deploy the people to earn revenue on other projects
- cut function and deliver a reduced system
- initiate an acceptance test before the software is debugged

All these strategies are effectively stopped by requiring the contractor to set out the high level development plan, the software size and provide the simple progress data described in Section 4, in this case every month.

6.1 Case Study 2 Results

Figure 6 below shows the actual defects reported from the development month by month. The QSM normalised defect model output is shown based on the contractor's most recent plan. This normalised model based on the plan is tuned using the actual reported data, the solid squares, to give the forecast of the outstanding defects shown by the open squares.

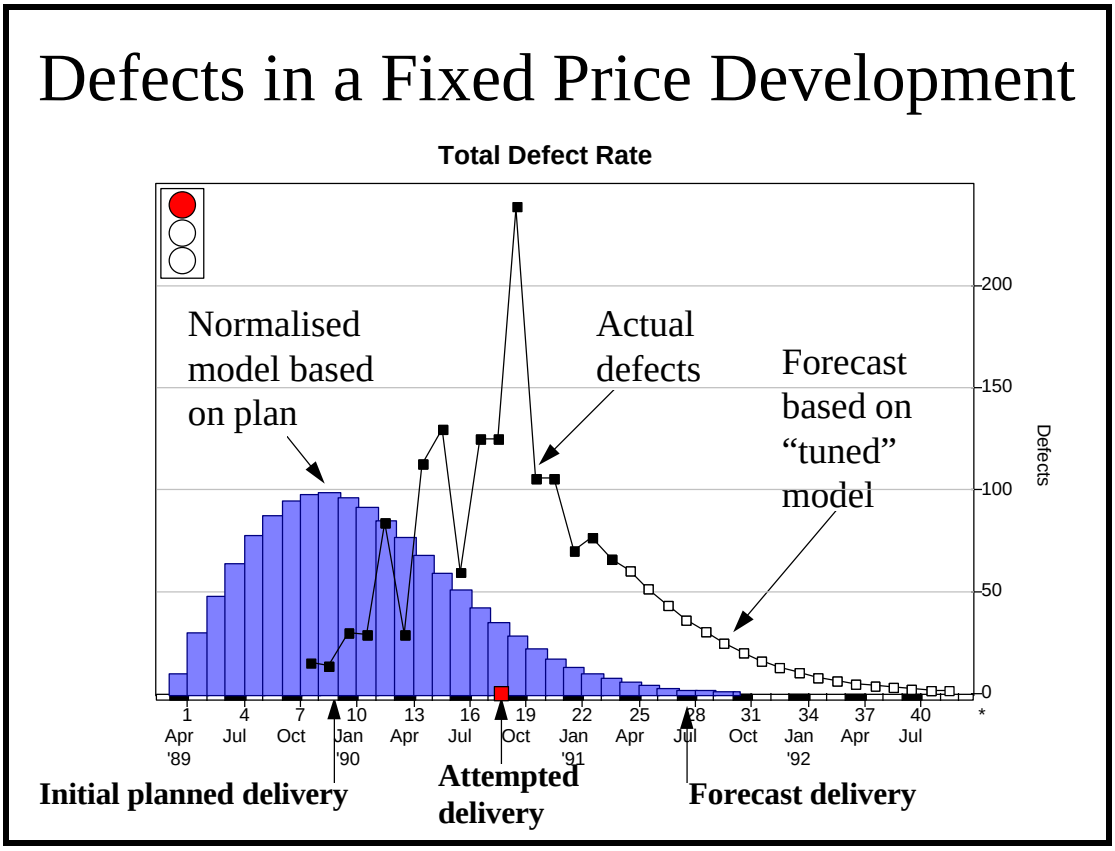

Figure 6 : Actual and Forecast Defects

An attempt was made by the contractor to persuade the client to accept the software in September 1990. This resulted in a surge of defects when independent tests began to be run. As a result the acceptance test was

terminated and the contractor required to continue testing to find and correct the defects.

The analysis in Figure 6 shows the situation in March 1991 when the forecast indicates that the software will be ready for acceptance in July 1991. Subsequently acceptance testing is expected to continue in order to improve the MTTD.

Overall this represents a slippage of 20 months with additional costs exceeding £3 million compared to the original contract. The case study highlights the significant commercial pressure on the contractor to minimise such large losses.

7. Observations

Reliability is a key attribute to the successful delivery of software. Simply focusing on effort and particularly time results in poor reliability with all the consequent dangers and costs of finding outstanding defects.

It is a high risk decision to deliver software without clear visibility of the error characteristics during development. We find the data necessary to give visibility is readily available as a normal part of development.

In all software development there is an integration and testing phase prior to delivery and operation. During this period the finding and fixing of errors is carried out. Staff responsible for finding the errors, usually separate from the programmers, keep logs recording each error including it's severity.

Error reports are passed to programmers who correct and report back their fix. After verifying the correction works the error log is updated. This means that the total errors found is available together with the number found each week or month. In addition the severity of each error permits the numbers in different categories (critical, serious, moderate and cosmetic) to be analysed.

It is practical to tune the error model by collecting the error data on a regular basis and using the result to forecast the outstanding errors. This in turn allows the date to be determined when the software is able to meet explicit reliability goals. The continuous tracking of errors then ensures the goals are reached before the software is delivered.

Attempts to prematurely deliver the software are forestalled. If a decision is made to deliver prematurely regardless of the reliability goals then the consequences are made clear to those responsible for such decisions.

8. Conclusions

Today successful software development means not only coming in on time and within budget but equally in achieving high reliability. This demands realistic estimates of development time, effort and defects using data and measures from a given development environment.

Informed purchasing organisations now require vendors to set out clearly :

- their high level software development staffing plan including key milestones
- for each sub-system the size estimates of software to be modified or developed
- use of existing off the shelf software (COTS)
- basic data from completed projects to enable the process productivity and quality of the vendor to be assessed

As the case studies illustrate this basic information is used contractually to track progress and determine reliability as the software development progresses. By quantitatively analysing this data a purchasing organisation is able to ensure that a vendor provides value for money and that software delivery takes place when the software meets explicit criteria for the time between errors.

In the same manner development organisations are able to forecast and monitor defects occurring in a given development. There is a clear need in an increasing number of application development areas to provide visibility and proof of the reliability of software before acceptance and operation.

Moreover the keeping and use of such data helps the development organisation to achieve higher levels of process maturity as defined by the Software Engineering Institute SEI. Work by QSM to quantify benefits from moving up the SEI maturity levels includes improvement to the Mean Time To Defect. Results show substantial benefits are measurable and demonstrable. (Ref. 6)

Ref. 1 : Royal Netherlands PTT Software Development Questionnaire Version 5 : Software Control Department
Ref. 2 : L.H. Putnam Measures For Excellence : Reliable Software, On Time, Within Budget Prentice Hall New York 1992
Ref. 3 : QSM Inc. Software Lifecycle Management Control, SLIM-Control Users Manual Special Topics : QSM Reliability Model
Ref. 4 : J.W.E. Greene The Software Control Office EC2 Software Engineering Conference Toulouse 1991
Ref. 5 : J.W.E. Greene Getting a Runaway Software Development under Control EC2 Software Engineering Conference Toulouse 1990
Ref. 6 : L.H. Putnam The Economic Value of Moving up the SEI Scale Software Technology Conference sponsored by the US Armed Forces, Salt Lake City, Utah 1993
Ref. 7 : S.H. Kan Modeling and Software Development Quality IBM Systems Journal Vol 30 No.3 1991